DETAILED ACTION

Continued Examination Under 37 CFR 1.114

1.   A request for continued examination under 37 CFR 1.114, including the fee set forth in

37 CFR 1.17(e), was filed in this application after final rejection. Since this application is

eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e)

has been timely paid, the finality of the previous Office action has been withdrawn pursuant to

37 CFR 1.114. Applicant's submission filed on 4/28/2011 has been entered.

2.   The reply filed 4/28/2011 has been fully considered. Claims 1, 7, 9, 16, 17, and 24 have

been amended. Claims 15, 23, and 25 have been canceled. Claims 1-4, 6-13, 16-21, 24, and 26-

37 are pending and have been examined.

Response to Arguments

3.   On pages 9-10 filed 4/28/2011, Applicant argues with respect to the 35 U.S.C. § 102(b)

rejection of claim 9. Similar arguments are provided on page 11 with respect to the rejection of

claim 1 over 35 U.S.C. § 103(a). It is noted that a new interpretation of the claim language is

applied in view of prior art of record U.S. Patent No. 5,193,191 (McKeeman et al.). As such,

Applicant's argument is moot.

4.   Applicant's arguments on pages 10-15 filed 11/17/2010 have been fully considered but

they are not persuasive.

        In response to applicant's argument on pages 11-12 that the examiner's conclusion of

obviousness is based upon improper hindsight reasoning, it must be recognized that any

judgment on obviousness is in a sense necessarily a reconstruction based upon hindsight

reasoning. But so long as it takes into account only knowledge which was within the level of ordinary skill at the time the claimed invention was made, and does not include knowledge gleaned only from the applicant's disclosure, such a reconstruction is proper. See *In re McLaughlin*, 443 F.2d 1392, 170 USPQ 209 (CCPA 1971).

In response to applicant's argument on page 12 that there is no teaching, suggestion, or motivation to combine the references, the examiner recognizes that obviousness may be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either in the references themselves or in the knowledge generally available to one of ordinary skill in the art. See *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988), *In re Jones*, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992), and *KSR International Co. v. Teleflex, Inc.*, 550 U.S. 398, 82 USPQ2d 1385 (2007). In this case, Robinson provides an explicit suggestion to combine by providing an indication of errors "in real time." This timely indication of errors provides a clear improvement that is faster and more efficient than compilers that do not work in the background.

Arguments on pages 14-15 with respect to the rejection of claim 34 under 35 U.S.C. § 103(a) regarding Piehler's suppression of error indication are not persuasive due at least to a reasonable broad interpretation of the claim language. Piehler teaches suppression of "at least one of an error and warning" (paragraph [0035]), and suppression of errors during execution (paragraph [0036]).

The remaining arguments are based upon previous arguments as addressed above, and are not persuasive for the same reasons.

Claim Rejections - 35 USC § 112

5.   The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

6.   Claims 26-35 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

7.   Claim 26 recites the limitation "claim 25" in line 1. There is insufficient antecedent basis for this limitation in the claim. For the purpose of further examination, this limitation will be interpreted as "claim 24". Claims 27-35 contain similar issues, or are dependent upon a claim with similar issues, as claim 26, and are similarly rejected and interpreted accordingly.

Claim Rejections - 35 USC § 102

8.   The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

9.   Claims 9-13 and 15 are rejected under 35 U.S.C. 102(b) as being anticipated by U.S. Patent No. 5,193,191 (McKeeman et al.).

As per claim 9, *McKeeman et al.* discloses an article comprising one or more machine-readable storage media containing instructions (see, e.g., col. 8, lines 6-39) that when executed enable a processor to:

initiate compiling of a file including one or more code segments (see, e.g., col. 11, lines 44-61 (recompilation uses previously compiled code);

detect a user request to compile the file (see, e.g., *Id.* (recompilation is started)); and

provide a result associated with the compiling in response to detecting the user request (see, e.g., col. 5, lines 23-34 (errors are detected and reported));

wherein the instructions when executed enable the processor to initiate compiling of the file based on determining that the file was modified (see, e.g., col. 5, lines 21-23).

wherein the instructions that when executed enable the processor to initiate compiling of the file based on determining that the file was modified comprise instructions that when executed enable the processor to indicate in a work queue that the file has been modified and to initiate compiling of the file in response to detecting the indication (see, e.g., Fig. 6b and associated text at col. 16, lines 54-63, describing a dependency analysis table used to identify modified files).

As per claim 10, *McKeeman et al.* further discloses the instructions when executed enable the processor to display a message to a user indicating that one or more errors were detected during the compiling (see, e.g., col. 5, lines 23-34 (errors are detected and reported)).

As per claim 11, *McKeeman et al.* further discloses the instructions when executed enable the processor to indicate to a user that the compiling was successful (see, e.g., col. 5, lines 23-34

(errors are detected and reported; if no errors are detected, then object code is produced as input

to the linker)).

As per claim 12, *McKeeman et al.* further discloses the instructions when executed enable

the processor to generate a file containing object code based on compiling the file and to store

the object code file in a temporary location (see, e.g., col. 5, lines 30-34).

As per claim 13, *McKeeman et al.* further discloses the instructions when executed enable

the processor to move the object code file from the temporary location into a product location

based on determining that the compiling of the file was successful and in response to detecting

the user request (see, e.g., col. 5, lines 23-34 (errors are detected and reported; if no errors are

detected, then object code is produced as input to the linker)).

As per claim 15, *McKeeman et al.* further discloses the instructions that when executed

enable the processor to initiate compiling of the file based on determining that the file was

modified comprise instructions that when executed enable the processor to indicate in a work

queue that the file has been modified and to initiate compiling of the file in response to detecting

the indication (see, e.g., col. 11, lines 44-61).

Claim Rejections - 35 USC § 103

10.   The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all

obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth
> in section 102 of this title, if the differences between the subject matter sought to be patented and the prior
> art are such that the subject matter as a whole would have been obvious at the time the invention was made
> to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be
> negatived by the manner in which the invention was made.

11.   Claims 1-4, 6-8, 16-21, 24, 26-29, 31-32, and 35-37 are rejected under 35 U.S.C. 103(a)

as being unpatentable over U.S. Patent No. 5,193,191 (McKeeman et al.) in view of "Upgrading

Microsoft Visual Basic 6.0 to Microsoft Visual Basic .NET" by (Robinson et al.).

As per claim 1, *McKeeman et al.* discloses:

initiating compilation of a file in a processor-based system … (see, e.g., col. 11, lines 44-

61 (recompilation uses previously compiled code));

McKeeman et al. does not expressly disclose: in advance of a request from a user to

compile the file. However, *Robinson et al.* teaches background compilation in advance of a

compilation request (see page 16 "Visual Basic .NET has a compiler that continually works in

the background.") It would have been obvious to one of ordinary skill in the art at the time the

invention was made to use the compilation of *McKeeman et al.* with the background compiler of

Robinson et al. in order to provide a timely indication of errors as suggested by *Robinson et al.*.

detecting the user request to compile the file (see, e.g., *Id.* (recompilation is started));

indicating a status of the compilation of the file in response to detecting the user request

(see, e.g., col. 5, lines 23-34 (errors are detected and reported)); and

wherein initiating compilation of the file comprises compiling the file in response to

determining that the file has been modified (see, e.g., col. 5, lines 21-23).

and wherein initiating compilation comprises: identifying the modified file in a work

queue; and initiating the compilation of the file based on the modified file being identified in the

work queue. (see, e.g., Fig. 6b and associated text at col. 16, lines 54-63, describing a

dependency analysis table used to identify modified files).

As per claim 2, *McKeeman et al.* further discloses initiating compilation of the file comprising compiling the file including one or more code segments to produce an object code file (see, e.g., col. 5, lines 30-34).

As per claim 3, *McKeeman et al.* further discloses compiling the file comprising compiling one or more code segments in the file to produce an object code file, and further comprising linking the object code file to produce an executable file (see, e.g., col. 5, lines 37-46), and wherein initiating compilation of a file in a processor-based system in advance of a request from a user to compile the file further comprises compiling the file to completion (see McKeeman Fig. 1 which depicts completion not only of object files, but also of an executable image; also see associated text in column 5).

As per claim 4, *McKeeman et al.* further discloses indicating the status of the compilation of the file comprising at least one of indicating that the compilation was successful and indicating that the compilation was unsuccessful (see, e.g., col. 5, lines 23-34).

As per claim 6, *McKeeman et al.* further discloses determining that the file has been modified comprising determining that the modified file has been saved to a storage unit (see, e.g., col. 5, lines 18-23).

As per claim 7, *McKeeman et al.* further discloses the file including one or more code segments (see, e.g., col. 3, lines 23-33).

As per claim 8, *McKeeman et al.* further discloses indicating the status of the compilation of the file comprising generating one or more files associated with the compilation of the file, storing the one or more generated files in a temporary location, and transferring the one or more

files from the temporary location to a different location in response to detecting the user request

(see, e.g., col. 5, lines 30-34).

As per claim 16, *McKeeman et al.* discloses an apparatus (see, e.g., col. 8, lines 6-39),

comprising:

a machine-readable storage medium containing means for initiating compilation of a file

in a processor-based system ...(see, e.g., col. 11, lines 44-61 (recompilation uses previously

compiled code));

... identifying the file as being modified in a work queue (see, e.g., Fig. 6b and associated

text at col. 16, lines 54-63, describing a dependency analysis table used to identify modified

files). *McKeeman* does not expressly disclose: in advance of a request from a user means for

identifying the file ... However, *Robinson et al.* teaches background compilation in advance of a

compilation request (see page 16 "Visual Basic .NET has a compiler that continually works in

the background."). It would have been obvious to one of ordinary skill in the art at the time the

invention was made to use the compilation of *McKeeman et al.* with the background compiler of

Robinson et al. to achieve the claimed invention. As disclosed in *Robinson et al.*, the motivation

for the combination would be to provide a timely indication of errors.

means for detecting the user request to compile the file (see, e.g., *Id.* (recompilation is

started)); and

means for indicating a status of the compilation of the file in response to detecting the

user request (see, e.g., col. 5, lines 23-34 (errors are detected and reported));

wherein the means for initiating compilation initiate compiling the file based on

determining that the file was modified and based on the modified file being identified in the

work queue (see, e.g., col. 5, lines 18-23; also Fig. 6b and associated text at col. 16, lines 54-63,

describing a dependency analysis table used to identify modified files).

McKeeman et al. further discloses a machine-readable storage media containing

instructions for implementing the recited functionality (see, e.g., col. 8, lines 6-39).

As per claim 17, *McKeeman et al.* discloses an apparatus (see, e.g., col. 8, lines 6-39),

comprising:

a storage unit having a file stored therein (see, e.g., col. 8, lines 6-39); and

a control unit communicatively coupled to the storage unit (see, e.g., col. 8, lines 6-39),

the control unit adapted to:

initiate compilation of the file (see, e.g., col. 11, lines 44-61 (recompilation uses

previously compiled code));

McKeeman does not expressly disclose: in advance of a request from a user to

compile the file. However, *Robinson et al.* teaches background compilation in advance

of a compilation request (see page 16 "Visual Basic .NET has a compiler that continually

works in the background.") It would have been obvious to one of ordinary skill in the art

at the time the invention was made to use the compilation of *McKeeman et al.* with the

background compiler of *Robinson et al.* in order to provide a timely indication of errors

as suggested by *Robinson et al.*.

detect the user request to compile the file (see, e.g., *Id.* (recompilation is started));

and

indicate a status of the compilation of the file in response to detecting the user request (see, e.g., col. 5, lines 23-34 (errors are detected and reported));

wherein the control unit is adapted to compile the file in response to determining that the file has been modified (see, e.g., col. 5, lines 18-23; col. 13, lines 42-54).

…and wherein the control unit adaptation to compile the file in response to determining that the file has been modified comprises: an adaptation to identify the modified file in a work queue; and an adaptation to initiate the processing of the file based on the modified file being identified in the work queue (see, e.g., Fig. 6b and associated text at col. 16, lines 54-63, describing a dependency analysis table used to identify modified files).

As per claim 18, *McKeeman et al.* discloses the control unit is adapted to compile a file including one or more code segments to produce an object code file (see, e.g., col. 5, lines 30-34).

As per claim 19, *McKeeman et al.* discloses the control unit is adapted to link the object code file to produce an executable file (see, e.g., col. 5, lines 37-46).

As per claim 20, *McKeeman et al.* discloses the control unit is adapted to store the executable file in a temporary location and to transfer the executable file from the temporary location to a different location based on detecting the user request (see, e.g., col. 5, lines 30-34).

As per claim 21, *McKeeman et al.* discloses the control unit is adapted to at least one of indicate that the compilation was successful and indicate that the compilation was unsuccessful (see, e.g., col. 5, lines 23-34).

As per claim 24, *McKeeman et al.* discloses:

identifying one or more source files that have been modified in a processor-based system and that the one or more source files are in a work queue (see, e.g., col. 5, lines 18-23; also Fig. 6b and associated text at col. 16, lines 54-63, describing a dependency analysis table used to identify modified files);

initiating processing of at least a portion of the modified source files (see, e.g., col. 11, lines 44-61);

McKeeman does not expressly disclose: before receiving a request to process the modified files. However, *Robinson et al.* teaches background compilation in advance of a compilation request (see page 16 "Visual Basic .NET has a compiler that continually works in the background.") It would have been obvious to one of ordinary skill in the art at the time the invention was made to use the compilation of *McKeeman et al.* with the background compiler of *Robinson et al.* in order to provide a timely indication of errors as suggested by *Robinson et al.*.

receiving the request to process at least one of the modified files (see, e.g., col. 11, lines 44-61 (recompilation uses previously compiled code)); and

providing a status associated with the processing of the file in response to receiving the request (see, e.g., col. 5, lines 23-34 (errors are detected and reported))…

…wherein the processor-based system is adapted to execute an integrated development environment module, wherein identifying the one or more files comprises the integrated development environment module placing the one or more of the source files that have been modified in a queue. (see, e.g., Fig. 6b and associated text at col. 16, lines 54-63, describing a dependency analysis table used to identify modified files).

As per claim 26, *McKeeman et al.* further discloses placing the one or more of the source files in the queue comprises placing at least one source file in the queue in response to a user saving the source file to a storage unit (see, e.g., col. 5, lines 18-23).

As per claim 27, *McKeeman et al.* further discloses placing the one or more of the source files in the queue comprises placing at least a portion of one source file in the queue in response to a user saving the source file to a storage unit using an editor and then exiting from the editor (see, e.g., col. 17, lines 11-36).

As per claim 28, *McKeeman et al.* further discloses placing the one or more of the source files in the queue comprises placing at least one source file in the queue in response to determining that a user desires to compile at least a portion of the source file as the source file is being edited (see, e.g., col. 5, lines 15-17; col. 11, lines 44-61).

As per claim 29, *McKeeman et al.* further discloses placing the one or more of the source files in the queue comprises placing at least one source file in the queue in response to determining that the source file includes at least one marker identifying a section of the source file that should be compiled, and wherein initiating processing of at least the portion of the one

or more modified files comprises compiling the identified section of the source file (see, e.g., col. 11, lines 44-51; also see, e.g., Fig. 6b and associated text at col. 16, lines 54-63).

As per claim 31, *McKeeman et al.* further discloses initiating the processing comprises initiating a build process to produce a software application and wherein receiving the request comprises receiving the request to building the software application (see, e.g., col. 5, lines 15-17).

As per claim 32, *McKeeman et al.* further discloses initiating the build process comprises performing compiling the modified source files to produce object code files and linking the object code files to produce executable files (see, e.g., col. 5, lines 37-46); and wherein initiating compiling further comprises compiling the file to completion (see McKeeman Fig. 1 which depicts completion not only of object files, but also of an executable image; also see associated text in column 5).

As per claim 35, *McKeeman et al.* further discloses the object code files and the executable files are moved to a different storage location in response to detecting the request and in response to detecting no error or warning (see, e.g., col. 5, lines 23-34 (errors are detected and reported; if no errors are detected, then object code is produced as input to the linker)).

As per claim 36, *McKeeman et al.* further discloses identifying one or more source files comprises identifying the one or more source files based on a directed acyclic graph (see, e.g., col. 16, 54-63).

As per claim 37, *McKeeman et al.* further discloses the directed acyclic graph includes a list of dependent files, wherein identifying one or more source files comprises identifying at least

one modified source file and another source file that is dependent on the modified source file

using the directed acyclic graph (see, e.g., col. 16, 54-63).

12.   Claim 30 is rejected under 35 U.S.C. 103(a) as being unpatentable over McKeeman et al.

and Robinson et al. as applied above, and further in view of U.S. Pat. App. Pub. No.

2005/0108682 (Piehler et al.).

        As per claim 30, *McKeeman et al.* discloses such a method (see the rejection of claims 24

and 25 under 35 U.S.C. § 102(b)) but fails to expressly disclose initiating the processing of the

modified source files comprises causing a background thread to awaken in response to placing

the one or more of the source files in the queue, where the background thread thereafter initiates

processing of the source files.   However, *Piehler et al.* teaches such use of a background thread

to enqueue a task for itself to complete the rest of the compilation in a background thread as part

of a response to a new file being added to a project or an existing file being modified (*Piehler et

al.* at paragraphs [0170] through [0180]. Therefore, it would have been obvious to include such

background thread use as per the teachings of *Piehler et al.* One would be motivated to do so to

gain the advantage of providing immediate feedback to the user without any detectable visible

delay in typing responsiveness while the compiler finishes processing the change (*Poehler et al.*

at paragraph [0174]).

13.   Claim 33 is rejected under 35 U.S.C. 103(a) as being unpatentable over McKeeman et al.

and Robinson et al. as applied above, and further in view of U.S. Pat. No. 6,230,313 (Callahan, II

et al.).

As per claim 33, *McKeeman et al.* further discloses placing the one or more of the source files in the queue comprising placing at least one source file in the queue in response to determining that the source file includes …[markers] identifying a section of the source file that should be compiled, wherein the … marker defines the beginning of the portion of the source file and the … marker defines the end of the portion, and wherein initiating processing of at least the portion of the one or more modified files comprises compiling the identified section of the source file (see, e.g., col. 16, line 60, through col. 17, line 6 (the compiler of *McKeeman* can identify changed portions (having a defined beginning and end) within a source code file and recompile only those changed portions)). *McKeeman* does not expressly disclose: at least two markers. However, *Callahan, II et al.* teaches the use of markers at the beginning and end of a region of interest (see column 10 lines 10-25). It would have been obvious to one of ordinary skill in the art at the time the invention was made to use McKeeman's markers with *Callahan*'s teaching of multiple markers in order to easily identify regions of interest as suggested by *Callahan*.

14.   Claim 34 is rejected under 35 U.S.C. 103(a) as being unpatentable over McKeeman et al. and Robinson et al. as applied above, and further in view of prior art of record U.S. Patent Application Publication US 2005/0114771 by Piehler et al. ("Piehler").

As per claim 34, *McKeeman et al.* discloses indication of errors (see, e.g., col. 5, lines 23-34 (errors are detected and reported)). McKeeman does not expressly disclose: suppressing at least one of an error and warning that is detected while compiling the modified source files. However, Piehler teaches suppression of error indication. See at least paragraphs [0034]-[0036].

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use McKeeman's error indication with Piehler's error suppression in order to make code completion more robust as suggested by Piehler (see paragraph [0035]).

Conclusion

15.   The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Matthew Gertz, "Advanced Basics; Scaling Up: The Very Busy Background Compiler," Microsoft Corp., MSDN Magazine, June 2005, 4 pages, discusses in more detail the implementation of the VB.NET background compiler and the changes in its design from the VISUAL BASIC .NET 2002 development environment to the VISUAL BASIC .NET 2005 development environment.

16.   Any inquiry concerning this communication or earlier communications from the examiner should be directed to JAMES D. RUTTEN whose telephone number is (571)272-3703. The examiner can normally be reached on M-F 10:00-6:30.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Li B. Zhen can be reached on (571)272-3768. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent

Application Information Retrieval (PAIR) system. Status information for published applications

may be obtained from either Private PAIR or Public PAIR. Status information for unpublished

applications is available through Private PAIR only. For more information about the PAIR

system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR

system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would

like assistance from a USPTO Customer Service Representative or access to the automated

information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/J. Derek Rutten/
Primary Examiner, Art Unit 2197